NAME

CLUSTER − cluster a table according to an index

SYNOPSIS

CLUSTER [(*option* [, ...])] [*table_name* [USING *index_name*]]

where *option* can be one of:

VERBOSE [*boolean*]

DESCRIPTION

CLUSTER instructs PostgreSQL to cluster the table specified by *table_name* based on the index specified by *index_name*. The index must already have been defined on *table_name*.

When a table is clustered, it is physically reordered based on the index information. Clustering is a one−time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's fillfactor storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

When a table is clustered, PostgreSQL remembers which index it was clustered by. The form **CLUSTER** *table_name* reclusters the table using the same index as before. You can also use the CLUSTER or SET WITHOUT CLUSTER forms of **ALTER TABLE** to set the index to be used for future cluster operations, or to clear any previous setting.

CLUSTER without a *table_name* reclusters all the previously−clustered tables in the current database that the calling user has privileges for. This form of **CLUSTER** cannot be executed inside a transaction block.

When a table is being clustered, an ACCESS EXCLUSIVE lock is acquired on it. This prevents any other database operations (both reads and writes) from operating on the table until the **CLUSTER** is finished.

PARAMETERS

table_name
: The name (possibly schema−qualified) of a table.

index_name
: The name of an index.

VERBOSE
: Prints a progress report as each table is clustered at INFO level.

boolean
: Specifies whether the selected option should be turned on or off. You can write TRUE, ON, or 1 to enable the option, and FALSE, OFF, or 0 to disable it. The *boolean* value can also be omitted, in which case TRUE is assumed.

NOTES

To cluster a table, one must have the MAINTAIN privilege on the table.

In cases where you are accessing single rows randomly within a table, the actual order of the data in the table is unimportant. However, if you tend to access some data more than others, and there is an index that groups them together, you will benefit from using **CLUSTER**. If you are requesting a range of indexed values from a table, or a single indexed value that has multiple rows that match, **CLUSTER** will help because once the index identifies the table page for the first row that matches, all other rows that match are probably already on the same table page, and so you save disk accesses and speed up the query.

CLUSTER can re−sort the table using either an index scan on the specified index, or (if the index is a b−tree) a sequential scan followed by sorting. It will attempt to choose the method that will be faster, based on planner cost parameters and available statistical information.

While **CLUSTER** is running, the search_path is temporarily changed to pg_catalog, pg_temp.

When an index scan is used, a temporary copy of the table is created that contains the table data in the index order. Temporary copies of each index on the table are created as well. Therefore, you need free space on disk at least equal to the sum of the table size and the index sizes.

When a sequential scan and sort is used, a temporary sort file is also created, so that the peak temporary space requirement is as much as double the table size, plus the index sizes. This method is often faster than the index scan method, but if the disk space requirement is intolerable, you can disable this choice by temporarily setting enable_sort to off.

It is advisable to set maintenance_work_mem to a reasonably large value (but not more than the amount of RAM you can dedicate to the **CLUSTER** operation) before clustering.

Because the planner records statistics about the ordering of tables, it is advisable to run **ANALYZE** on the newly clustered table. Otherwise, the planner might make poor choices of query plans.

Because **CLUSTER** remembers which indexes are clustered, one can cluster the tables one wants clustered manually the first time, then set up a periodic maintenance script that executes **CLUSTER** without any parameters, so that the desired tables are periodically reclustered.

Each backend running **CLUSTER** will report its progress in the pg_stat_progress_cluster view. See Section 27.4.2 for details.

Clustering a partitioned table clusters each of its partitions using the partition of the specified partitioned index. When clustering a partitioned table, the index may not be omitted. **CLUSTER** on a partitioned table cannot be executed inside a transaction block.

EXAMPLES

Cluster the table employees on the basis of its index employees_ind:

```
CLUSTER employees USING employees_ind;
```

Cluster the employees table using the same index that was used before:

```
CLUSTER employees;
```

Cluster all tables in the database that have previously been clustered:

```
CLUSTER;
```

COMPATIBILITY

There is no **CLUSTER** statement in the SQL standard.

The following syntax was used before PostgreSQL 17 and is still supported:

```
CLUSTER [ VERBOSE ] [ table_name [ USING index_name ] ]
```

The following syntax was used before PostgreSQL 8.3 and is still supported:

```
CLUSTER index_name ON table_name
```

SEE ALSO

clusterdb(1), Section 27.4.2